Methods for Run-Time Load Management in Distributed Simulation Systems

Greg Schow, STRICOM
Rhonda Freeman, TASC
Anthony Lashley, TASC
Jeff Swauger, TASC
12443 Research Parkway
Orlando, FL 32826

KEYWORDS
RTI, load balancing

ABSTRACT
Large numbers of simulation systems and network assets are required to generate the number of entities needed for large scale distributed simulation exercises. These assets are both costly and often in limited supply, particularly high bandwidth networks and simulation centers. As a result, the number of hardware platforms and network connections required for an exercise are often held to a minimum, which results in computational platforms and network hardware being loaded at higher levels than would ordinarily be considered optimum, with less excess capacity available to respond to unexpected changes in the simulation scenario. A method of providing a dynamic load balancing capability to redistribute network and computational loads in real-time over the entire networked simulation system to preclude such occurrences is required. This paper proposes an organized approach to the problem of dynamic load balancing and arrive at a set of algorithms and applications which offer the potential for significant advances in real-time load balancing performance. The use of predictive algorithms based on network load monitoring, use of pre-exercise analyses to generate "heuristic triggers" to be used during run-time, expert systems based on human operator experiences, and intelligent agents based on fuzzy logic artificial intelligence approaches will all be discussed.

1.0 Problem Description

Large scale High Level Architecture (HLA) distributed simulation exercises require numerous high value hardware assets to achieve the large number of entities and complex interactions required. These assets include the computational systems which generate and control the various aspects of the simulation and the network used to provide communications between the various systems. Computational systems are used for a variety of purposes from simulation of entities (including behavior) in constructive simulations, man in the loop simulations, C4I systems, and weather and other synthetic environments. In order to minimize costs, the number of hardware platforms and network connections must be kept to a minimum, which requires efficient use of all simulation assets. In general, this results in computational platforms and network hardware being loaded at higher levels than might be optimum, with less excess capacity available to respond to the varying demands of the exercise. A balance between the number of entities per hardware platform and required network bandwidth for multiple platforms must be maintained. If the network and platform loading in a simulation exercise could be maintained at a constant level the minimum amount of simulation resources required could be easily determined. Unfortunately the nature of simulation exercises precludes this level of certainty. The computational and network loading varies during the course of the exercise as a result of the interactions between the various simulated entities. With larger and larger simulation exercises involving increasing numbers of entities, platform and network loading are always important concerns. In the past, loading problems have been dealt with mostly in the exercise planning phase, however a simulation exercise is a highly dynamic event, and no amount of planning can accommodate all possible engagements. The number of entities a single platform can efficiently represent depends greatly on the current level of engagement. Even with very thorough load balancing planning, unexpected situations where network and simulation platform bottlenecks appear will surface. Dynamic load balancing for distributed simulation systems is therefore an area in which focused research can be expected to contribute significant gains in performance.

In previous efforts, the use of Simulation of Simulation (Sim of Sim) applications have provided valuable insight into potential bottlenecks and performance problems. In addition, some real-time network and simulation platform load monitoring capabilities have been developed (such as those being developed under the DISECT project for incorporation into STOW 97), however both of these have required human participation and decision making to operate. The increasing complexity of large simulation exercises makes it difficult to impossible for a human operator to monitor and react to all facets of an exercise. Network loading, simulation platform loading, and scenario development must all be continuously evaluated and monitored.

As a result of the uncertainties present in large scale distributed simulation exercises, a method for distributing the network and computational loads during real time is required. Unbalanced loads waste computational and network resources and can lead to decreased performance, unrealistic simulation results, and even failure of the simulation exercise and network. The goal of load balancing is to distribute simulation processing among the resources involved in a manner that minimizes both load imbalances and communications between the systems. Load balancing must be designed to have a high degree of transparency for the systems, i.e. the act of load balancing must not overburden the system and cause the very type of problems it is intended to address.

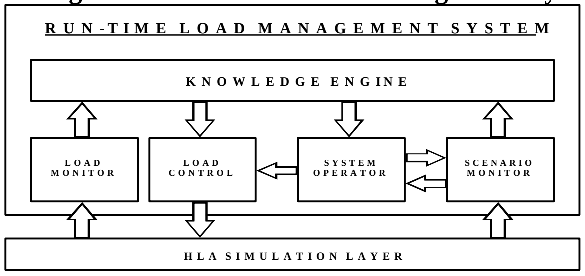
While there has been a considerable amount of theoretical work done on the subject of load balancing in the area of parallel and distributed computing, most of this work assumes an a priori knowledge of the computations to be performed and the resources available. Implementing a flexible and adaptive real time load balancing system will require an integrated, highly automated system with sufficient intelligence to predict network and simulation platform loading and react proactively before problems are encountered. The goal of this paper is to identify promising approaches to the problem of dynamic load balancing.

SUBMISSION Technical Approach

System Architecture:

Prior experience gained in the development of load and network monitoring applications for the STOW 97 program have been used to determine a high level system architecture and functional breakdown of the dynamic load balancing problem. Figure 1 illustrates the architecture proposed for study and implementation. The internal architecture of the run-time load management system consists of six main functional areas as described below.

Figure 1. Run-Time Load Management System



Interface with the Simulation: As can be seen, the run-time load management system interfaces with the HLA simulation via the RTI. Information regarding the state of the simulation and commands to the system are passed through the RTI. It is possible that implementation of an effective system may require the ability to acquire information independent of the RTI, such as monitoring of processor and network loads via polling of individual computers, network nodes and devices. Ideally the RTI will be used for all interfaces, although at early stages of development the RTI may not support all functions required.

Load Monitoring: The load monitoring function includes all applications and methods used to acquire information about the loading of the simulation exercise, including information regarding the status and load of the network as well as the status of individual computational assets participating in the exercise. A significant part of the proposed effort will be to identify those metrics that provide useful and accurate information concerning the load of the simulation system. Each application will likely require tailoring or calibration of the load monitoring approach to arrive at a metric that reflects the actual loading of the system. As an example, ModSAF has been modified for the STOW 97 exercise to allow the load of each ModSAF station to be evaluated in real time. For this application, a simple metric such as CPU loading is inadequate, as ModSAF is designed to use essentially all of the CPU cycle. Therefore, even an idle ModSAF station simulating no entities or interactions will appear to be fully loaded based on CPU utilization. ModSAF is being monitored using a modification that allows the frame rate of the simulation to be read by the load monitor. As loading increases on a ModSAF station, the simulation frame rate will decrease from the normal, unburdened 15 Hz rate. At some frame rate the station will "gasp," indicating that it is nearing critical loading. Monitoring of the frame rate allows the status to be monitored so that load balancing can be triggered prior to this event.

Load Control: The load control function implements the actual commands to the simulation exercise (both network and simulation systems) to implement redistribution of simulation entities and processes to prevent overload conditions. The exact nature of these commands will be determined by the capabilities of the systems and network involved in the simulation exercise. Ownership management is an approach that has been used successfully in load balancing efforts, and the foundation from which this research will build. In the case of ModSAF, it is possible to transfer ownership of an entity or entities from one station to another, shifting the computational load from highly burdened systems to low use systems. For networks, dynamic multicast groups and routers may be capable of being directed to alter the flow of information to preclude overloading of key network nodes.

System Operator: The system operator is a human in the loop who makes decisions regarding the information on system loading and appropriate load balancing responses. The main goal of this effort will be to minimize and eventually eliminate the need for human intervention.

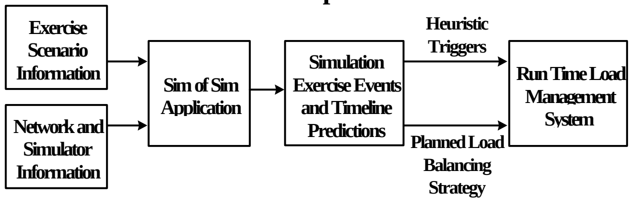
Scenario Monitor: The scenario monitor evaluates the progress of the simulation exercise relative to that predicted by pre-exercise Sim of Sim analyses. Based on the correlation between the progression of the actual exercise and that predicted this function is capable of flagging upcoming overload conditions with sufficient time to implement load balancing before difficulties arise. The scenario monitor can also provide warning that the simulation exercise is diverging from what was predicted, and that prior assumptions based on pre-exercise analysis are no longer valid.

Knowledge Engine: The knowledge engine is an intelligent agent based system which monitors the status of the simulation exercise and makes decisions and recommendations concerning load balancing. This function is the focus of the majority of the effort of this task. It is expected that this function will start out being an assistant to help a human operator make rapid decisions and implement appropriate actions. As the nature of the load balancing problem and it's relationship to the system loading metrics being evaluated increases, the goal is to mature this application to the point where little or no human intervention is required.

Dynamic Load Balancing Algorithm Development

The dynamic load balancing algorithm development task will result in the processes that are implemented in the knowledge engine described above. Figure 2 illustrates a functional flow diagram of the predicted tasks related to generation of load balancing parameters for a particular exercise. It is expected that the exact nature of the algorithms used and the implementation chosen will be highly dependent on the simulation exercise to be performed and the hardware and software assets that make up the simulation framework.

Figure 2. Dynamic Load Balancing Algorithm Development


Exercise Scenario Information
Network and Simulator Information
Sim of Sim Application
Simulation Exercise Events and Timeline Predictions
Heuristic Triggers
Planned Load Balancing Strategy
Run Time Load Management System

Information regarding the exercise scenario and simulation assets are used to perform a Sim of Sim analysis of the simulation exercise. The results of the Sim of Sim analysis consist of information regarding predicted timelines, actions, system and network loading, and potential problem areas that are expected to occur in the simulation exercise. Various Sim of Sim approaches will be evaluated, with the accuracy of the predictions compared with the results of simulations run on real networked distributed simulation exercises. This approach will be used to refine the Sim of Sim application to increase its accuracy in predicting simulation results. The results of the Sim of Sim for an exercise will be used to generate information to be used by the dynamic load balancing application in monitoring and evaluating simulation exercise status. While a major goal of this effort is to arrive at generic load balancing approaches that are useful in a wide variety of situations, it is predicted that optimum results will not be obtained from such non-exercise specific approaches. The results of the Sim of Sim will be used to tailor load balancing strategies and generate "heuristic triggers" to be used to by the load balancing algorithm. Heuristic triggers are events or trends that are predicted to precede occurrence of network or computer overload for the planned exercise. These heuristic triggers allow the load balancing algorithm to better recognize and predict upcoming problems based on warning signs that are related to the specific exercise involved. As a result of these experiments, better generic predictive approaches to the load balancing problem will be possible.

3.0 For The Future

There are a wide variety of features, or policies, related to dynamic load balancing that will be explored in the development of these load balancing approaches, including:

1. Transfer Policy - What conditions and events will trigger load balancing?

2. Initiation Policy - Where will the load balancing be initiated? Will there be one central site that controls load balancing for the entire system, or will multiple distributed nodes be used? Do overloaded systems attempt to offload tasks, or do under-utilized systems attempt to acquire tasks?

3. Selection Policy - Which tasks will be transferred between systems? This will be dependent on the types of systems used in the simulation exercise.

4. Location Policy - When load balancing occurs, which system or node will the tasks be transferred to? How will this be determined?

5. Cooperation Policy - What options do each system or node have with respect to load balancing? Does the receiving node have a choice as to whether to accept the tasks or is the receiver forced to take on tasks regardless of its internal evaluation of its load?

The use of expert systems and fuzzy logic based intelligent algorithms will be investigated to provide this function. It is expected that a fuzzy logic based approach will prove to be the most promising technology for dynamic load balancing due to its flexibility and adaptability to conditions that are not easily predicted or defined.